

Dynamic Spyware Analysis

M. Egele¹ & C. Kruegel¹ & E. Kirda¹ &
H. Yin^{2,3} & D. Song²

¹Secure Systems Lab
Vienna University of Technology

²Carnegie Mellon University

³College of William and Mary

USENIX Annual Technical Conference, June 21, 2007

spyware - a threat to internet users

- Spyware is malware that is installed on a computer to monitor user actions
- Spyware is an important threat to the security and privacy of Internet users
 - An analysis by Webroot and Earthlink showed that a large portion of Internet-connected computers are infected with spyware
 - Spyware also degrades performance and causes unexpected side-effects
- BHOs are a very popular kind of spyware (Weng et al, 90 of 120 spyware samples use BHO architecture)

spyware - a threat to internet users

- Spyware is malware that is installed on a computer to monitor user actions
- Spyware is an important threat to the security and privacy of Internet users
 - An analysis by Webroot and Earthlink showed that a large portion of Internet-connected computers are infected with spyware
 - Spyware also degrades performance and causes unexpected side-effects
- BHOs are a very popular kind of spyware (Weng et al, 90 of 120 spyware samples use BHO architecture)

spyware - a threat to internet users

- Spyware is malware that is installed on a computer to monitor user actions
- Spyware is an important threat to the security and privacy of Internet users
 - An analysis by Webroot and Earthlink showed that a large portion of Internet-connected computers are infected with spyware
 - Spyware also degrades performance and causes unexpected side-effects
- BHOs are a very popular kind of spyware (Weng et al, 90 of 120 spyware samples use BHO architecture)

spyware - a threat to internet users

- Spyware is malware that is installed on a computer to monitor user actions
- Spyware is an important threat to the security and privacy of Internet users
 - An analysis by Webroot and Earthlink showed that a large portion of Internet-connected computers are infected with spyware
 - Spyware also degrades performance and causes unexpected side-effects
- BHOs are a very popular kind of spyware (Weng et al, 90 of 120 spyware samples use BHO architecture)

spyware - a threat to internet users

- Spyware is malware that is installed on a computer to monitor user actions
- Spyware is an important threat to the security and privacy of Internet users
 - An analysis by Webroot and Earthlink showed that a large portion of Internet-connected computers are infected with spyware
 - Spyware also degrades performance and causes unexpected side-effects
- BHOs are a very popular kind of spyware (Weng et al, 90 of 120 spyware samples use BHO architecture)

drawbacks of existing signature-based tools

A number of signature-based anti-spyware products exist that share some drawbacks of that approach

- Unable to detect previously unknown threats
- Need continuous signature updates
- Often require human analysis before creating signatures

drawbacks of existing signature-based tools

A number of signature-based anti-spyware products exist that share some drawbacks of that approach

- Unable to detect previously unknown threats
- Need continuous signature updates
- Often require human analysis before creating signatures

drawbacks of existing signature-based tools

A number of signature-based anti-spyware products exist that share some drawbacks of that approach

- Unable to detect previously unknown threats
- Need continuous signature updates
- Often require human analysis before creating signatures

behavior-based detection

To overcome the shortcomings of signature-based detectors

- We implemented a behavioral-based detection technique
- That classifies a program as spyware if
 - It monitors user behavior
 - And then leaks the gathered information to a third party (the attacker)

behavior-based detection

To overcome the shortcomings of signature-based detectors

- We implemented a behavioral-based detection technique
- That classifies a program as spyware if
 - 1 It monitors user behavior
 - 2 And then leaks the gathered information to a third party (the attacker)

behavior-based detection

To overcome the shortcomings of signature-based detectors

- We implemented a behavioral-based detection technique
- That classifies a program as spyware if
 - 1 It monitors user behavior
 - 2 And then leaks the gathered information to a third party (the attacker)

behavior-based detection

To overcome the shortcomings of signature-based detectors

- We implemented a behavioral-based detection technique
- That classifies a program as spyware if
 - 1 It monitors user behavior
 - 2 And then leaks the gathered information to a third party (the attacker)

our approach

- Focus of analysis on BHOs
- We use dynamic analysis to monitor BHO for presence of malicious behavior
- Two challenges need to be solved
 - Track the flow of sensitive data throughout the system
 - Observe what actions are performed by the BHO under analysis

our approach

- Focus of analysis on BHOs
- We use dynamic analysis to monitor BHO for presence of malicious behavior
- Two challenges need to be solved
 - Track the flow of sensitive data throughout the system
 - Observe what actions are performed by the BHO under analysis

our approach

- Focus of analysis on BHOs
- We use dynamic analysis to monitor BHO for presence of malicious behavior
- Two challenges need to be solved
 - 1 Track the flow of sensitive data throughout the system
 - 2 Observe what actions are performed by the BHO under analysis

our approach

- Focus of analysis on BHOs
- We use dynamic analysis to monitor BHO for presence of malicious behavior
- Two challenges need to be solved
 - 1 Track the flow of sensitive data throughout the system
 - 2 Observe what actions are performed by the BHO under analysis

our approach

- Focus of analysis on BHOs
- We use dynamic analysis to monitor BHO for presence of malicious behavior
- Two challenges need to be solved
 - 1 Track the flow of sensitive data throughout the system
 - 2 Observe what actions are performed by the BHO under analysis

our approach

Our solution features three key components

- 1 URLs and page contents considered to contain sensitive information
- 2 The propagation of this data throughout the system is observed by taint tracking
- 3 By monitoring system calls, attempts of leaking sensitive information can be identified

our approach

Our solution features three key components

- 1 URLs and page contents considered to contain sensitive information
- 2 The propagation of this data throughout the system is observed by taint tracking
- 3 By monitoring system calls, attempts of leaking sensitive information can be identified

our approach

Our solution features three key components

- 1 URLs and page contents considered to contain sensitive information
- 2 The propagation of this data throughout the system is observed by taint tracking
- 3 By monitoring system calls, attempts of leaking sensitive information can be identified

analysis output

The system classifies BHOs as spyware or legitimate software.
Additionally comprehensive reports

- File actions
- Network actions
- Interprocess communication
- Operating system actions
- Enrich the reports with more details when sensitive data is involved

analysis output

The system classifies BHOs as spyware or legitimate software.
Additionally comprehensive reports

- File actions
- Network actions
- Interprocess communication
- Operating system actions
- Enrich the reports with more details when sensitive data is involved

analysis output

The system classifies BHOs as spyware or legitimate software.
Additionally comprehensive reports

- File actions
- Network actions
- Interprocess communication
- Operating system actions
- Enrich the reports with more details when sensitive data is involved

analysis output

The system classifies BHOs as spyware or legitimate software.
Additionally comprehensive reports

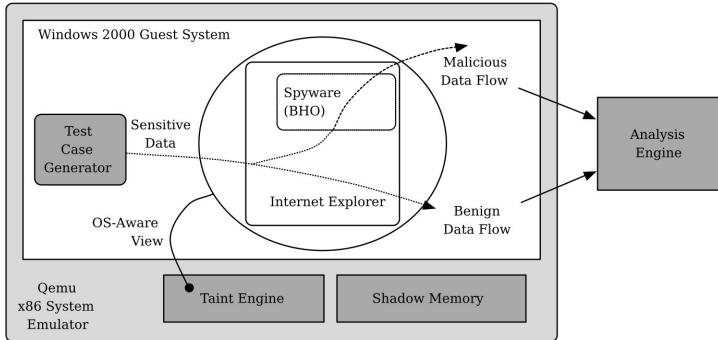
- File actions
- Network actions
- Interprocess communication
- Operating system actions
- Enrich the reports with more details when sensitive data is involved

analysis output

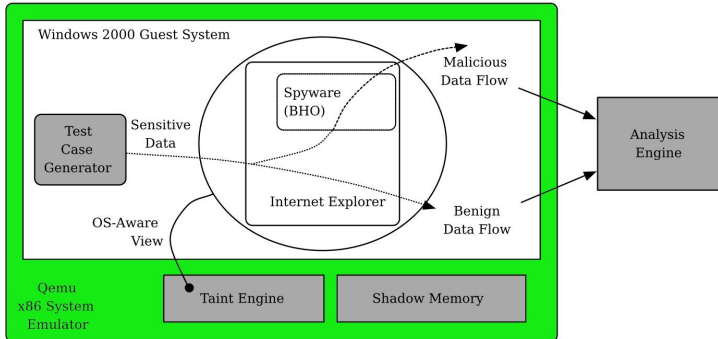
The system classifies BHOs as spyware or legitimate software.
Additionally comprehensive reports

- File actions
- Network actions
- Interprocess communication
- Operating system actions
- Enrich the reports with more details when sensitive data is involved

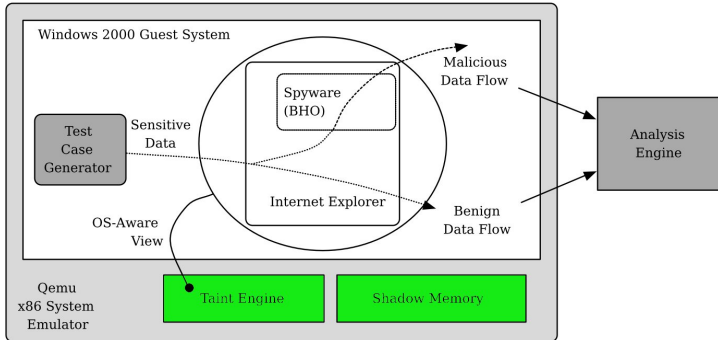
system overview



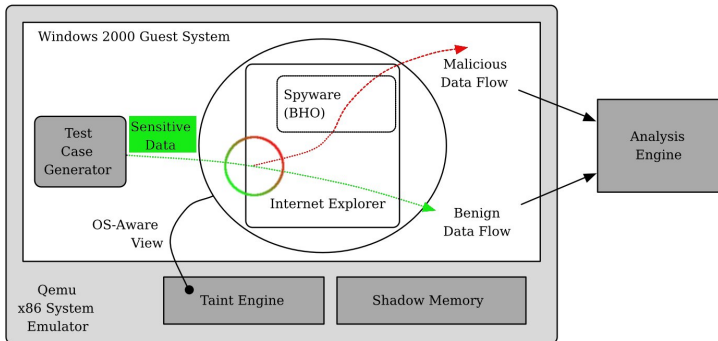
system overview



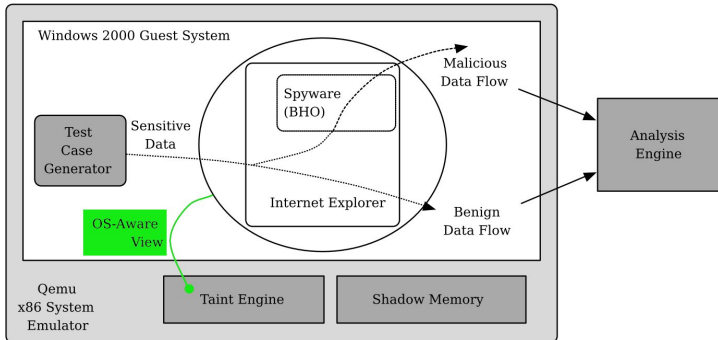
system overview



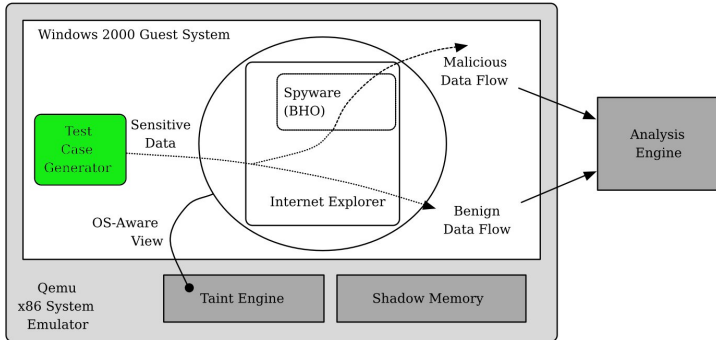
system overview



system overview



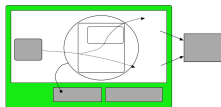
system overview



qemu x86 system emulator

Qemu was enhanced to perform additional tasks

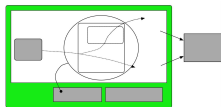
- Perform taint tracking operations
- Provide hooking capabilities for system / function calls
- Monitoring capabilities for system actions



qemu x86 system emulator

Qemu was enhanced to perform additional tasks

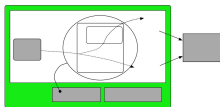
- Perform taint tracking operations
- Provide hooking capabilities for system / function calls
- Monitoring capabilities for system actions



qemu x86 system emulator

Qemu was enhanced to perform additional tasks

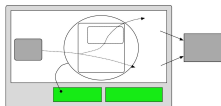
- Perform taint tracking operations
- Provide hooking capabilities for system / function calls
- Monitoring capabilities for system actions



taint tracking

Tainting allows to tag data elements of interest and track their propagation throughout the system. Our system covers

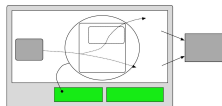
- Data dependencies (`mov %eax, %ebx`)
- Address dependencies
(`mov %eax, (%ebx, 2, 1)`)
- Control dependencies
(`if (x == 'a') {y = 'a'}`)
- Untainting with simple constant functions
(`xor %eax, %eax`)



taint tracking

Tainting allows to tag data elements of interest and track their propagation throughout the system. Our system covers

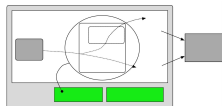
- Data dependencies (`mov %eax, %ebx`)
- Address dependencies
(`mov %eax, (%ebx, 2, 1)`)
- Control dependencies
(`if (x == 'a') {y = 'a'}`)
- Untainting with simple constant functions
(`xor %eax, %eax`)



taint tracking

Tainting allows to tag data elements of interest and track their propagation throughout the system. Our system covers

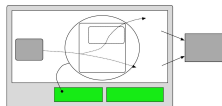
- Data dependencies (`mov %eax, %ebx`)
- Address dependencies
(`mov %eax, (%ebx, 2, 1)`)
- **Control dependencies**
(`if (x == 'a') {y = 'a'}`)
- Untainting with simple constant functions
(`xor %eax, %eax`)



taint tracking

Tainting allows to tag data elements of interest and track their propagation throughout the system. Our system covers

- Data dependencies (`mov %eax, %ebx`)
- Address dependencies
(`mov %eax, (%ebx, 2, 1)`)
- **Control dependencies**
(`if (x == 'a') {y = 'a'}`)
- Untainting with simple constant functions
(`xor %eax, %eax`)



control dependencies

While data and address dependencies can be handled on a per instruction basis, control dependencies cannot. Instead

- Whenever a branch depends on tainted data, the scope for this branch is calculated (static analysis)
- In this scope, targets of all write operations are tainted (independent of taint status of operands)
- After scope ends, normal taint operations resume
- It is possible that independent variables become tainted, but no false positives were observed in our experiments

control dependencies

While data and address dependencies can be handled on a per instruction basis, control dependencies cannot. Instead

- Whenever a branch depends on tainted data, the scope for this branch is calculated (static analysis)
- In this scope, targets of all write operations are tainted (independent of taint status of operands)
- After scope ends, normal taint operations resume
- It is possible that independent variables become tainted, but no false positives were observed in our experiments

control dependencies

While data and address dependencies can be handled on a per instruction basis, control dependencies cannot. Instead

- Whenever a branch depends on tainted data, the scope for this branch is calculated (static analysis)
- In this scope, targets of all write operations are tainted (independent of taint status of operands)
- After scope ends, normal taint operations resume
- It is possible that independent variables become tainted, but no false positives were observed in our experiments

control dependencies

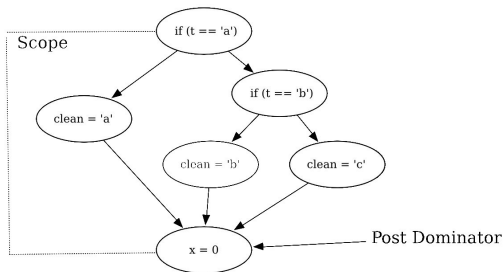
While data and address dependencies can be handled on a per instruction basis, control dependencies cannot. Instead

- Whenever a branch depends on tainted data, the scope for this branch is calculated (static analysis)
- In this scope, targets of all write operations are tainted (independent of taint status of operands)
- After scope ends, normal taint operations resume
- It is possible that independent variables become tainted, but no false positives were observed in our experiments

cover control dependencies

Assume variable `t` is tainted

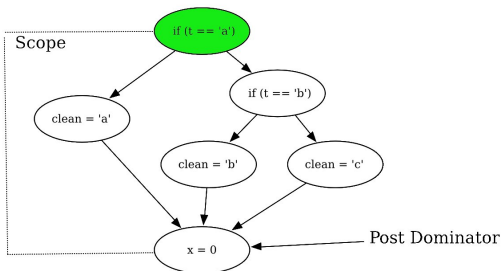
```
if (t == 'a')  
  clean = 'a';  
else {  
  if (t == 'b')  
    clean = 'b';  
  else  
    clean = 'c';  
}  
x = 0;
```



cover control dependencies

Assume variable `t` is tainted

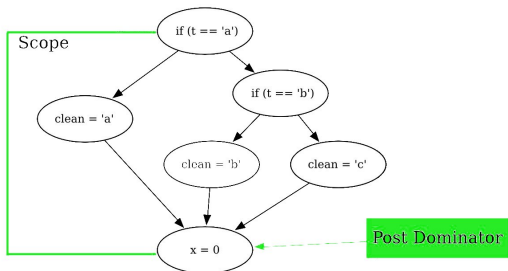
```
if (t == 'a')
  clean = 'a';
else {
  if (t == 'b')
    clean = 'b';
  else
    clean = 'c';
}
x = 0;
```



cover control dependencies

Assume variable `t` is tainted

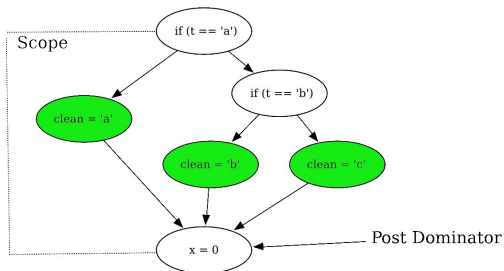
```
if (t == 'a')  
  clean = 'a';  
else {  
  if (t == 'b')  
    clean = 'b';  
  else  
    clean = 'c';  
}  
x = 0;
```



cover control dependencies

Assume variable `t` is tainted

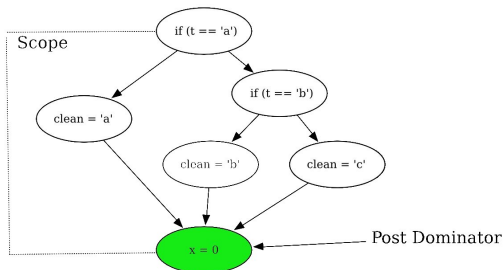
```
if (t == 'a')  
  clean = 'a';  
else {  
  if (t == 'b')  
    clean = 'b';  
  else  
    clean = 'c';  
}  
x = 0;
```



cover control dependencies

Assume variable `t` is tainted

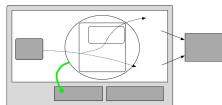
```
if (t == 'a')  
  clean = 'a';  
else {  
  if (t == 'b')  
    clean = 'b';  
  else  
    clean = 'c';  
}  
x = 0;
```



bridging the semantic gap

Some key tasks have to be performed to connect operating system information with hardware level taint information

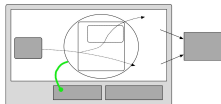
- Identify the currently executing task/thread
- Check if the current instruction is executed in the context of the BHO
- Monitor operating system actions (task/thread switches) and system calls (creation of new processes, ...)



bridging the semantic gap

Some key tasks have to be performed to connect operating system information with hardware level taint information

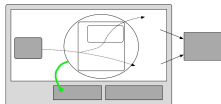
- Identify the currently executing task/thread
- Check if the current instruction is executed in the context of the BHO
- Monitor operating system actions (task/thread switches) and system calls (creation of new processes, ...)



bridging the semantic gap

Some key tasks have to be performed to connect operating system information with hardware level taint information

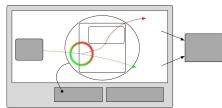
- Identify the currently executing task/thread
- Check if the current instruction is executed in the context of the BHO
- Monitor operating system actions (task/thread switches) and system calls (creation of new processes, ...)



taint sources

A taint source defines a portion of data that is sensitive. Two taint sources have been implemented so far

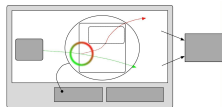
- The URL that is loaded by the Internet Explorer (`IWebBrowser2::Navigate()`)
- Contents of network packages received by the Internet Explorer over TCP connections (`NtDeviceIoControlFile`)



taint sources

A taint source defines a portion of data that is sensitive. Two taint sources have been implemented so far

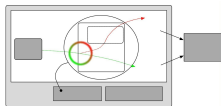
- The URL that is loaded by the Internet Explorer (`IWebBrowser2::Navigate()`)
- Contents of network packages received by the Internet Explorer over TCP connections (`NtDeviceIoControlFile`)



taint sinks

Taint sinks are parts of the system that are of interest when receiving tainted data. So far, we have taint sinks for the following actions

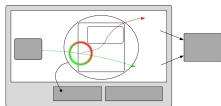
- Writing to a file
(including memory mapped files)
- Writing to the registry
- Writing to network sockets (tcp/udp)
- Writing to shared memory regions
(i.e., for interprocess communication)
- Certain asm instructions
(i.e., (string-)compares)



taint sinks

Taint sinks are parts of the system that are of interest when receiving tainted data. So far, we have taint sinks for the following actions

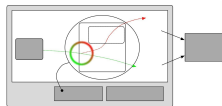
- Writing to a file
(including memory mapped files)
- Writing to the registry
- Writing to network sockets (tcp/udp)
- Writing to shared memory regions
(i.e., for interprocess communication)
- Certain asm instructions
(i.e., (string-)compares)



taint sinks

Taint sinks are parts of the system that are of interest when receiving tainted data. So far, we have taint sinks for the following actions

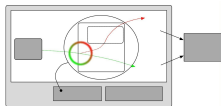
- Writing to a file
(including memory mapped files)
- Writing to the registry
- Writing to network sockets (tcp/udp)
- Writing to shared memory regions
(i.e., for interprocess communication)
- Certain asm instructions
(i.e., (string-)compares)



taint sinks

Taint sinks are parts of the system that are of interest when receiving tainted data. So far, we have taint sinks for the following actions

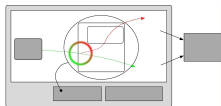
- Writing to a file
(including memory mapped files)
- Writing to the registry
- Writing to network sockets (tcp/udp)
- Writing to shared memory regions
(i.e., for interprocess communication)
- Certain asm instructions
(i.e., (string-)compares)



taint sinks

Taint sinks are parts of the system that are of interest when receiving tainted data. So far, we have taint sinks for the following actions

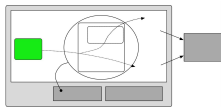
- Writing to a file
(including memory mapped files)
- Writing to the registry
- Writing to network sockets (tcp/udp)
- Writing to shared memory regions
(i.e., for interprocess communication)
- Certain asm instructions
(i.e., (string-)compares)



automated analysis of BHOs

For batch-analysis of multiple BHOs we implemented an automated testing tool

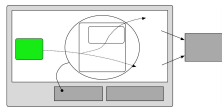
- First, the browser session of a user is captured (different kinds of web-sites are visited)
- For every BHO in the batch, replay the captured session and perform the analysis



automated analysis of BHOs

For batch-analysis of multiple BHOs we implemented an automated testing tool

- First, the browser session of a user is captured (different kinds of web-sites are visited)
- For every BHO in the batch, replay the captured session and perform the analysis



the bare numbers

Results for analysis

	Spyware	False Negative	Benign	Suspicious	False Positive	Total
Spyware	21	0	-	-	-	
Benign	-	-	12	1	1	14

Different mechanisms used by spyware to leak sensitive data

Network	File System	Registry	Shared Memory	Total
11	1	3	6	21

Automated crawling and analysis ongoing (millions of URLs, hundreds of samples)

the bare numbers

Results for analysis

	Spyware	False Negative	Benign	Suspicious	False Positive	Total
Spyware	21	0	-	-	-	
Benign	-	-	12	1	1	14

Different mechanisms used by spyware to leak sensitive data

Network	File System	Registry	Shared Memory	Total
11	1	3	6	21

Automated crawling and analysis ongoing (millions of URLs, hundreds of samples)

the bare numbers

Results for analysis

	Spyware	False Negative	Benign	Suspicious	False Positive	Total
Spyware	21	0	-	-	-	
Benign	-	-	12	1	1	14

Different mechanisms used by spyware to leak sensitive data

Network	File System	Registry	Shared Memory	Total
11	1	3	6	21

Automated crawling and analysis ongoing (millions of URLs, hundreds of samples)

the bare numbers

Results for analysis

	Spyware	False Negative	Benign	Suspicious	False Positive	Total
Spyware	21	0	-	-	-	
Benign	-	-	12	1	1	14

Different mechanisms used by spyware to leak sensitive data

Network	File System	Registry	Shared Memory	Total
11	1	3	6	21

Automated crawling and analysis ongoing (millions of URLs, hundreds of samples)

details on false positive and suspicious samples

- One false positive: PrivacyBird
request the w3c/p3p.xml document from same server
- One suspicious sample: LostGoggles
request JavaScript file with referrer set to the visited URL

details on false positive and suspicious samples

- One false positive: PrivacyBird
request the w3c/p3p.xml document from same server
- One suspicious sample: LostGoggles
request JavaScript file with referrer set to the visited URL

details on spyware samples

- `zangohook.dll` reads the current URL and copies it to a shared memory that is then read by `Zango.exe` “companion process”
- `e2give` reads the URL of every site and compares it to a list (automatically detected) of URLs stored in the BHO; upon a match, the request is redirected to a different server with the original URL as a parameter
- `stup.dll` submits the URLs of all visited pages to a remote server. This BHO is not detected by the latest versions (at the time of writing) of AdAware or SpyBot.

details on spyware samples

- zangohook.dll reads the current URL and copies it to a shared memory that is then read by Zango.exe “companion process”
- e2give reads the URL of every site and compares it to a list (automatically detected) of URLs stored in the BHO; upon a match, the request is redirected to a different server with the original URL as a parameter
- stup.dll submits the URLs of all visited pages to a remote server. This BHO is not detected by the latest versions (at the time of writing) of AdAware or SpyBot.

details on spyware samples

- zangohook.dll reads the current URL and copies it to a shared memory that is then read by Zango.exe “companion process”
- e2give reads the URL of every site and compares it to a list (automatically detected) of URLs stored in the BHO; upon a match, the request is redirected to a different server with the original URL as a parameter
- stup.dll submits the URLs of all visited pages to a remote server. This BHO is not detected by the latest versions (at the time of writing) of AdAware or SpyBot.

summary

- Taint-tracking-based, behavioral spyware analysis
- Focus is on BHOs
- Covers data-, address-, and control dependencies
- Able to detect previously unknown spyware instances

summary

- Taint-tracking-based, behavioral spyware analysis
- Focus is on BHOs
- Covers data-, address-, and control dependencies
- Able to detect previously unknown spyware instances

summary

- Taint-tracking-based, behavioral spyware analysis
- Focus is on BHOs
- Covers data-, address-, and control dependencies
- Able to detect previously unknown spyware instances

summary

- Taint-tracking-based, behavioral spyware analysis
- Focus is on BHOs
- Covers data-, address-, and control dependencies
- Able to detect previously unknown spyware instances

Thank You

Questions?