No Loitering: Exploiting Lingering Vulnerabilities in Default COM Objects

David Dewey
dewey@us.ibm.com

Patrick Traynor
traynor@cc.gatech.edu
Introduction
COM Background – Introduction

- Language-neutral design philosophy for the creation of components for MS Windows
- Defines a base framework for creating plugins and components for myriad MS products
- Plugins are identified by a GUID or class id (CLSID) stored in the Windows Registry
- Allows for the persistence of object state between instantiations
COM Background – Security

- Enforcing the instantiation of third-party objects is a significant security concern
- Currently, few discrete applications enforce a black list
- CLSID’s are checked at instantiation time against a list in the Windows Registry
- As vulnerabilities are discovered in COM objects, they are often just listed in the killbit list – not actually fixed
COM Background – Management

- Average Windows install will have 1000’s of COM Objects
- Current killbit list has over 600 entries
- Many libraries contain multiple COM objects
Vulnerability Characterization – Architectural Weakness

- Black lists are only enforced on controls loaded by the base executable itself
- Trusted COM objects may load any other object – without security verification
- By creating a specially crafted persistence stream, one COM object can be coerced into loading another
Vulnerability Characterization – Attack Requirements

An attacker must have the following:

1) An application that will render adversary-controlled content
   Internet Explorer, MS Word, MS Excel, Adobe Reader, etc.

2) An application that will load COM objects
   Internet Explorer, MS Word, MS Excel, Adobe Reader, etc.

3) A COM object that will in turn load other COM objects
   Many objects that are based on the MS ATL

4) A vulnerable object that can be exploited
   Killbit list is has over 600 entries
Vulnerability Characterization – Proof of Concept

1) Create a MS Word Document – can be emailed, rendered by browser, etc.
2) COM objects can be embedding in Word right through the GUI
3) Load MS Date and Time picker control
4) Have the control above load Microsoft Helper Object for Java
5) Exploit vulnerability in Helper Object for Java
Vulnerability Characterization – Breadth of Attack

- Many applications allow instantiation of COM objects
  - MS Office, Adobe Reader, Internet Explorer, Flash, etc.

- A new application could be created today

- Trying to secure each application individually is a fool’s errand
Mitigation Architecture – Assumptions

- Our goal is to prevent the exploitation of this vulnerability on a clean system
- We intend to adhere to Microsoft’s design model
- We do not intend to protect infected systems
- We do not intend to protect against the instantiation of COM objects by malicious COM containers

- BOTTOM LINE: We intend to stop the initial attack
Mitigation Architecture – High-Level Architecture

- Hook every COM instantiation API
- Look up the CLSID in a pre-defined black list
- Terminate the instantiation as necessary
Results and Discussion – Effectiveness

- Successfully stopped attacks against:
  - MS Internet Explorer
  - MS Word
  - MS Excel
  - MS PowerPoint
  - “Homemade” COM Container
  - ActiveX Control Test Container
Results and Discussion – Performance

- Average lookup time of 554µs
  - 95% confidence interval of ±104µs

- Using Microsoft API’s to query registry
  - Linear scan of registry

- Could be improved with a more intelligent database
Results and Discussion – Policy Creation

- Working from the killbit list is still difficult
- Experimented with creating per-application lists
- Experimented with deploying system-wide – interesting side effects
Results and Discussion – Practical Impact

- **Microsoft Security Vulnerabilities**
  - MS10-083
  - MS10-036
  - MS09-060
  - MS09-037
  - MS09-035

- **Disclosed through US-CERT VU #456745**
  - Adobe APSB09-10
  - Cisco-SA-20090728
  - F5 Networks FirePass Controls
  - SonicWALL XTSAC.cab
  - Sun Alert 264648
Conclusion

- How many gates do you have to put up?
- Standard COM architecture creates a transitive trust issue
- Many COM containers on the average Windows install
- Hundreds of vulnerable COM object lingering on the average Windows install
- Windows needs a centralized solution for the management of COM security
Questions?

dewey@us.ibm.com
traynor@cc.gatech.edu