Prof. Claudio Enrique Righetti
Departamento de Computación – Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Pabellón I – Ciudad Universitaria – (C1428EGA) – Buenos Aires – Argentina
Tel./Fax:+54-11-4576-3359 - claudio@dc.uba.ar

OAT N° 03/15
Auditoría de sistemas Elecciones 2015 Ciudad de Buenos Aires

INFORME FINAL

Resumen ejecutivo

El presente informe finaliza el proceso de auditoría iniciado el viernes 20 de febrero con la firma de la Orden de Asistencia Técnica entre el Tribunal Superior de Justicia y la Facultad de Ciencias Exactas y Naturales de la Universidad de Buenos Aires.

El mismo complementa los cinco informes presentados los días 16 de abril; 1, 9 y 23 de junio y 17 de julio y contiene las conclusiones de toda la auditoría y observaciones sobre el proceso eleccionario.

El informe está estructurado en tres partes, en la primera se realiza un resumen de las principales tareas y observaciones realizadas durante el desarrollo de la auditoría del sistema de votación y escrutinio utilizado en las elecciones de la Ciudad de Buenos Aires del año 2015. En la segunda parte se presentan algunos aspectos a analizar en un Sistema de votación y escrutinio.

Por último se presentan recomendaciones para futuras elecciones en lo relacionado con la utilización de tecnología en la emisión, recuento y totalización de votos. Estas recomendaciones son aplicables en distintos momentos de la preparación del sistema electoral, empezando por el marco normativo, el establecimiento de los requerimientos deseables, la redacción de los pliegos de licitación (en caso de hacerse con un proveedor privado), la forma de encarar el desarrollo, la auditoría, las pruebas y la puesta en funcionamiento.

Auditoría del Sistema de Votación y Escrutinio

Insumos para la auditoría

La auditoría se realizó en base al Anexo II de la Ley 4894 y la sección 3, "Pliego de Especificaciones Técnicas" del Pliego de Bases y Condiciones publicado como Anexo de la Resolución N° 21/MJYSGC/15 del Ministerio de Justicia y Seguridad.

El trabajo de auditoría se basó en la observación del hardware de la máquina de votación, del código fuente de la aplicación, la documentación y las reuniones que se tuvieron con las partes.

Se destaca la falta de un *Manual de Operaciones*, que debería incluir todos los aspectos de configuración, instalación, despliegue, uso, mantenimiento, repliegue y obtención y publicación de resultados.

Desarrollo de la auditoría

Procedimientos y soporte documental

La auditoría del sistema incluyó el análisis de los procedimientos asociados que junto a la funcionalidad del sistema (las máquinas de voto y recuento, las máquinas de transmisión, el soporte documental) en relación a los requisitos pedidos por la ley y la comparación con la votación en papel.

Se destaca de lo analizado que la posibilidad de cumplir con los requisitos de la ley está principalmente sostenido en los procedimientos y el soporte documental del voto y de los certificados de escrutinio de mesa. Es importante mantener en el futuro estas características (u otras equivalentes):

1. que el propio votante pueda controlar el registro de su opción, utilizando un medio físico que se puede verificar independientemente del sistema informático (soporte documental, también llamado boleta única electrónica), que él mismo introduce en la urna (confirmación física) y que tiene visible en forma impresa el texto completo de su opción (para corroborar nombres de candidatos, nombres de partidos, agrupaciones o alianzas y número de lista antes de introducir su voto en la urna);
2. que los fiscales puedan verificar el inicio del acto electoral "*en cero*" (con la urna vacía), el comienzo del recuento "*en cero*" (con la visualización permanente de todos los contadores), el progreso del recuento (siguiendo el procedimiento definido de la lectura por parte de la autoridad de mesa de la boleta que se está contando, la visualización de la misma y la visualización permanente de todos los contadores, con la iluminación de la última posición que ha sido contabilizada en cada categoría);
3. que los fiscales puedan llevarse un certificado de escrutinio emitido por la misma máquina que emite el Acta de Cierre y Escrutinio (de modo de asegurarse que no hay errores de transcripción entre ellos);
4. que los partidos políticos tengan acceso en forma oportuna a la información digitalizada del escrutinio de todas las mesas escrutadas en el escrutinio provisorio, hayan estado presentes o no en cada mesa o escuela.

Los puntos mencionados anteriormente apuntan a garantizar la transparencia del resultado y el respeto de la voluntad del votante donde el principal protagonista de esa transparencia es el soporte documental. Otros aspectos son más subjetivos: para garantizar que se respete el deseo del votante, tiene que poder encontrar fácilmente al candidato que desea votar; eso se analizará en la sección funcionalidad. Con respecto a la privacidad del voto, existiendo tecnología dedicada a desarrollar dispositivos espías, es imposible garantizar la protección perfecta y completa (tanto para el voto con dispositivos electrónicos como con el sistema tradicional que solo usa papel y sobre); también se analizará ese aspecto más adelante.

En los informes anteriores se hizo hincapié en la necesidad de definir y seguir procedimientos específicos durante el acto eleccionario y en la capacitación, tanto de las autoridades de mesa como de la ciudadanía en general.

Funcionalidad

La auditoría de los aspectos funcionales del software consideró como tales a las características del mismo que son visibles por los usuarios o que determinan qué es lo que el usuario puede hacer y cómo.

Esta auditoría entiende como usuarios del software a:

- Los ciudadanos en general en su rol de electores y autoridades de mesa en las máquinas de votación.
- Las autoridades electorales que deben poder tomar decisiones sobre ciertos aspectos del proceso, así como sus delegados en los sitios de votación.
- Los partidos políticos en su rol de fiscales.

Se analizó si el sistema de votación y escrutinio, en forma clara y eficiente, permitía a los usuarios realizar sus tareas y si proveía de mecanismos de control para minimizar los errores provenientes del uso del mismo.

A lo largo de todo el proceso de Auditoría se han realizado observaciones sobre aspectos funcionales del sistema, algunas de las cuales han sido atendidas en versiones posteriores del software, como ha sido detallado en los distintos informes presentados.

Se presentan algunas de las mejoras realizadas por la empresa en función de las observaciones realizadas por la auditoría que mejoraron las características funcionales del software:

1. Se presentaron aclaraciones, recordatorios y otras ayudas en las pantallas para el elector que faciliten el uso del sistema.

2. Se aumentó el tamaño de la tipografía de los textos impresos en los distintos soportes documentales para facilitar la lectura por parte del elector al verificar su voto, y de las autoridades de mesa y fiscales al realizar el escrutinio.
3. Se agregó en la boleta un cartel señalando la posición del chip RFID con la leyenda "Verifique su voto" para recordarle al elector que lo haga y señalándole la ubicación.
4. En base a recomendaciones del Tribunal, y luego de conversaciones mantenidas entre el equipo de auditores y la empresa, esta última introdujo una modificación en el funcionamiento del software de la máquina de votación en lo relativo al acceso que se logra con la credencial de técnico. Luego de esta modificación, el técnico sólo puede cambiar el estado de la máquina con la participación de la autoridad de mesa.

Se presentan algunas de las observaciones que se han mantenido a los largo de todo el proceso de auditoría y que recomendamos evaluar en futuros requerimientos funcionales de un sistema de votación y escrutinio:

1. El sistema presenta en orden aleatorio las listas cada vez que el elector ingresa a la pantalla donde se muestran las listas. Si un elector pasa varias veces por esta pantalla, el orden de la misma se presenta distinto cada vez. Esto podría confundir al ciudadano que recorra las pantallas más de una vez.

   La solución propuesta por esta auditoría fue que el sorteo del orden de las listas se realice cada vez que ingresa un nuevo elector en el sistema (cuando se inserta una BUE en blanco) y se mantenga fijo hasta que ese elector haya confirmado toda su selección (hasta que imprima o retire la BUE). La solución implementada consistió en mostrar un recuadro azul sobre la elección realizada, de modo que si el elector vuelve a la pantalla, aunque la posición se vuelve a sortear, queda claro lo último que seleccionó[1].

   El impacto de este proceso de disposición aleatoria fue relativo en la primera vuelta y despreciable en la segunda dado que la cantidad de agrupaciones que llegaron a competir fue relativamente baja (seis o siete en la primera vuelta, obviamente dos en la segunda). Si la cantidad de agrupaciones que compiten es grande (como suele ocurrir en las PASO), el problema del cambio de orden en una misma sesión puede ser realmente confuso para el elector.
2. En el modo de "Votación Asistida" disponible para personas con dificultades visuales, si el votante presiona un número pero olvida presionar la tecla "#" el sistema no le advierte de esa situación y se queda esperando indefinidamente.

1 eso presentaba una dificultad adicional porque podía confundir dando a entender que la selección ya se había hecho; la forma de avanzar era volver a elegir un candidato (que por supuesto podía volver a ser el ya elegido que se mantenía marcado con el recuadro azul)

Luego de un tiempo de espera, el sistema debería volver a indicar las instrucciones necesarias para llevar a cabo la elección, o indicarle nuevamente que debe presionar la tecla "#" a continuación del número.

3. Al seleccionar "Versión de Demostración" para utilización en la capacitación no hay ningún indicio visual que muestre que se está en este modo y no en el modo de impresión de votos reales. El "modo de demostración" debería tener *siempre* un indicio visual en la pantalla, más allá de que se usen candidaturas reales o ficticias.

Código fuente auditado

El código fuente auditado tiene una gran cantidad de debilidades de distintos tipos. Puede a veces parecer (si se analiza el código fuente como un elemento estático que no cambia en el tiempo) que señalar debilidades es exagerado o de poca utilidad; sin embargo, en esta auditoría, se observaron una cantidad de cambios en el código fuente entre la primera y la segunda vuelta mucho mayor a lo esperado, si se piensa que sólo estaba planificado cambiar las categorías a elegirse (una sola en vez de tres con dos listas en vez de seis o siete) y el diseño de la pantalla.

Para algunas observaciones en particular, que son indicios de clases de observaciones, remitimos al lector a los Anexos titulados "Observaciones sobre el código fuente" en los informes 1, 2, 4 y 5.

En general, lo que se ha observado es que el código fuente no sigue pautas de programación segura, ni hay una metodología o conjunto de convenciones unificadas en la escritura de las distintas partes del código. También se observaron partes duplicadas de código que se podrían haber evitado con técnicas de programación estructurada, programación orientada a objetos, programación orientada a aspectos o programación guiada por intenciones.

En el Anexo I se analizan distintas características que mejorarían el código y su auditabilidad, las mismas están basadas en las debilidades observadas durante el proceso de auditoría.

Procedimiento de instalación del software de la máquina de voto

La configuración y generación del DVD con el software para la máquina de voto lo hizo la empresa utilizando un procedimiento que no estaba documentado ni fue auditado[2] que incluía modificar el código fuente; la información recolectada para dicha configuración (nombres de las listas y de los candidatos, fotos de los candidatos principales, información sobre mesas, comunas y su ubicación, fonética

2 sí se auditó el resultado final (o sea el DVD listo para insertar en la máquina de voto), pero no el procedimiento aplicado; en la sección recomendaciones incluimos una mención a este tema.

de las lecturas de todos estos ítems para el modo asistido para ciegos) provino de fuentes diversas y no tuvo una instancia unificada de verificación por parte de las autoridades previa a la generación de los DVD[3].

Asimismo, se deben prever los protocolos de copia y resguardo de los DVDs que se utilizarán durante los comicios al estilo de los definidos en los Anexos de las Resoluciones 138 y 142 del TSJ.

Transmisión de datos y seguridad

Entre ambas elecciones (primera y segunda vuelta) se observaron cambios en la forma de encarar la seguridad y los procedimientos de transmisión. En particular se utilizaron técnicas de "*security by obscurity*", que pueden haber estado justificadas como medida de emergencia porque fueron tomadas a último momento. También cambió el protocolo de transmisión de claves y certificados de seguridad de puntos de transmisión entre lo planificado originalmente y los dos operativos.

Asimismo, la empresa no utilizó el mecanismo de transmisión de datos de contingencia utilizando los códigos QR de los Certificados de Transmisión y teléfonos celulares con cámaras aun en los casos en los que hubo inconvenientes para transmitir utilizando internet o la red de datos móvil, pese a que dicho mecanismo formaba parte de los mecanismos de contingencia previstos en la documentación provista por la empresa a esta auditoría.

El mecanismo de generación y distribución de claves, identificadores, certificados, tokens, etc es de difícil implementación. Son muchos los factores a tener en cuenta, la oportunidad de la distribución, el canal de distribución, la identificación del receptor, el error humano, el bloqueo o reemplazo en caso de necesidad, etc. Más adelante recomendamos hacer un análisis de riesgos; parte de ese análisis debe ser previo a la selección de una tecnología y debe incluir los aspectos de generación, distribución y administración de claves, identificadores, tokens, certificados, etc.

Se observó que los procesos que corrían dentro de la máquina de voto lo hacían con permisos de "`root`" (administrador o superusuario), esto es riesgoso ya que ante una vulnerabilidad que afecte a cualquier parte del sistema, si se la logra explotar, es trivial que el atacante obtenga el control total del equipo. Esto se evita poniendo a cada proceso a correr con los mínimos privilegios necesarios.

También conviene mejorar la seguridad eliminando del DVD para la máquina de voto paquetes del sistema operativo que no son imprescindibles (manejadores de paquetes, utilidades de administración de usuarios, editores de texto, shells interactivos, compiladores, etc).

3 la verificación se hizo utilizando máquinas de votación con DVDs de arranque generados, verificando toda esa información manualmente comuna por comuna.

Hardware - máquina de voto

Se destaca respecto de las máquinas de voto la facilidad de instalación, y su diseño (de plástico sin aristas ni vértices), tamaño y peso; lo que facilita el traslado y almacenamiento de las mismas.

Parte de las vulnerabilidades detectadas podían ser controladas en base a la custodia de la máquina de voto y de su tapa de acceso a los puertos y a la lectora de DVD. Eso determinó que en los procedimientos recomendados en el Anexo II del Informe N° 2 se hiciera hincapié en la posición de la máquina (visible para la autoridad de mesa durante todo el acto electoral), y se indicara especial atención a la tapa. La tapa no tenía precintos, llaves ni candados[4], el control era solo visual. Se recomienda evaluar las vulnerabilidades detectadas de manera que se facilite el control de la tapa de algún modo más notable, como una señal lumínica y/o sonora que advierta de la apertura de la misma.

Aspectos a analizar en un Sistema de Votación y escrutinio

El sistema de votación no es solo el software que corre en una máquina de votación. Cuando se analiza un sistema de votación es importante revisar en conjunto (en forma sistémica) todos los aspectos que lo conforman. En especial:

- la definición del sistema (marco legal, requerimientos, especificaciones, etc),
- la máquina de votación (el hardware y firmware de todos sus componentes internos),
- el sistema informático (software con sus aspectos funcionales y su código fuente) y el sistema operativo elegido,
- los procedimiento definidos,
- los aspectos de seguridad implementados,
- los planes de contingencia
- la documentación, planes de desarrollo, mapas, definición de arquitectura,
- los planes de testing, control de calidad, etc

4 haberlos tenido habría agregado mayor complejidad a los protocolos, ya sea por la distribución de llaves o claves o por el procedimiento a aplicar ante una eventual violación de un *sticker* de seguridad.

Sistema Terminado – Separación de la configuración y el código

El objetivo de un sistema es ser una herramienta que sea útil para determinada tarea específica. Podemos decir que un programa está terminado cuando, sin la necesidad de agregar líneas de código fuente al programa, éste puede ser usado por el cliente[5] según la especificación de la herramienta[6]. En el caso de un sistema para una máquina de votar diseñado para las elecciones de la Ciudad debe permitir ser utilizado tanto en las PASO, como en la primera vuelta, como en la eventual segunda vuelta sin necesidad de modificar ninguna línea del programa. Todo lo que cambia entre esas tres instancias corresponde a la configuración y esta configuración debe poder realizarla el cliente.

Tiene que haber un rol de "*usuario administrador*" o "*usuario configurador*" que sea el que, con la capacitación y documentación necesaria, pueda cargar los metadatos de la configuración de modo de poder utilizar el sistema en cualquiera de los escenarios posibles. La estructura de los metadatos y las pantallas de configuración deben cumplir los criterios de usabilidad estándar y tener mecanismos de control para evitar errores de transcripción. La carga de metadatos debe permitir tanto la carga masiva de tablas enteras como el cambio de registros individuales. El "*usuario administrador*" tiene que ser capaz de configurar el sistema y generar los DVD o los instaladores de las máquinas de voto sin la participación de la empresa proveedora del sistema.

Aspectos funcionales

Un sistema de votación y escrutinio es usado por una amplia variedad de usuarios con diferentes características como edades, estratos sociales, habilidades y competencias. Por lo tanto la usabilidad del software adquiere una gran importancia en el desarrollo de un sistema informático de votación y escrutinio.

Si el sistema permite hacer la tarea lo más rápidamente posible, si es amigable y ayuda a que el usuario cometa menos errores o se recupere de ellos fácilmente, y si además el usuario queda satisfecho con su uso, se puede considerar que el sistema tiene una buena usabilidad.

El sistema tiene que ser fácil de usar, fácil de aprender, intuitivo así como simple en todas las etapas del proceso en las que se usa.

---

5 en este caso el cliente es la autoridad electoral o autoridad de aplicación o el organismo encargado de la organización de los comicios.

6 que un programa esté terminado no significa que esté cerrado a futuras modificaciones no previstas o a mejoras o adaptaciones a nuevas resoluciones o cambio en la legislación; que esté terminado significa que con lo que ya está se puede usar para todas las instancias previstas con la legislación vigente al momento de entregar el programa.

En la etapa de emisión de voto un elector con una mínima instrucción debe poder entender cómo elegir para cada categoría el candidato que desea votar, cómo modificar su elección, cómo emitir su voto y cómo verificarlo. En la etapa de escrutinio el ciudadano que cumple el rol de autoridad de mesa debe poder entender, también con una mínima instrucción, cómo realizar las distintas tareas que tiene bajo su responsabilidad. Esto es fundamental porque ayuda a la confiabilidad y transparencia del proceso por parte de los ciudadanos. Entender qué se hace minimiza la incertidumbre y la desconfianza.

Análisis de riesgos

Todo sistema tiene riesgos potenciales, ya sea que utilice o no dispositivos informáticos. El análisis de riesgos debe hacerse en comparación con las distintas alternativas, incluidos los sistemas manuales de voto y escrutinio con papel.

Respecto al riesgo de no respetar la voluntad del votante, un sistema informatizado que base sus operaciones en el soporte documental (como el caso de la BUE) tiene riesgos equivalentes a los sistemas en papel.

Respecto al riesgo de no respetar la privacidad del votante (secreto del voto), en todos los sistemas con papel o con uso de equipos informáticos el riesgo existe. Cuando se utilizan cuartos oscuros (obligatorio para la votación en papel) algún elector podría introducir una cámara que filme y transmita en directo o grabe. Cuando se utilizan máquinas no hay manera de garantizar que no se les haya introducido un dispositivo espía que haga un registro digital de cada voto en el orden en que fueron hechos (y quizás con la hora también). Si bien la introducción de estos elementos es más difícil (en especial si la máquina está custodiada todo el tiempo), de lograrse introducir un elemento tal, el daño es mayor porque el registro puede ser procesado digitalmente.

En una situación similar a la usada en los comicios auditados donde el control de los padrones sea manual y no se registre el orden en que cada ciudadano vota, para que tales registros tengan alguna utilidad deben combinarse con una persona que esté registrando las identidades de los votantes y el orden en que lo hacen.

Si la votación utilizando dispositivos electrónicos se combina con el control electrónico de la identidad de los votantes en el padrón electoral se correría el riesgo de que alguien altere ambos sistema para obtener registros ordenados de votantes y de votos, y luego pueda cruzar esa información en forma digital vulnerando en forma masiva la privacidad de los votantes. Por ello en la próxima sección recomendaremos hacer por un lado un análisis de riesgos y por el otro no utilizar simultáneamente en el mismo acto electoral sistemas informáticos para emitir el voto y para controlar la identidad de los votantes.

Recomendaciones

A continuación referimos una serie de recomendaciones a seguir en el futuro si se fuere a continuar utilizando sistemas informáticos en diversas etapas del proceso eleccionario, en particular, durante la emisión del voto y la transmisión al centro de cómputos.

Sobre las especificaciones, los requerimientos y la elección del sistema

1. Recomendamos que las autoridades que conduzcan el proceso de votación (Poder Ejecutivo, Tribunal Superior de Justicia o quien corresponda) encaren con tiempo suficiente el proceso de incorporación o elección de tecnologías en la aplicación del voto, que las tareas anteriores a la selección de la solución estén listas **al menos un año antes del comicio** para poder hacer las auditorías, pruebas y correcciones necesarias. De esa manera la auditoría puede recomendar acciones o correcciones que pueden implementarse sin arriesgar la estabilidad del sistema, dejando tiempo suficiente para volver a probar las correcciones.
2. Recomendamos que los requerimientos sean lo más detallados posible (incluyendo requerimientos funcionales, de procedimientos, de seguridad, etc) y que estén disponibles al menos dos meses antes del llamado a licitación, de modo de poder auditar si los requerimientos se adecuan a la normativa vigente y si permiten obtener una solución completa y segura.
3. Recomendamos requerir al proveedor del servicio que defina exhaustivamente las especificaciones del software y hardware que provee.
4. Recomendamos que el código fuente sea público de manera que no sólo tengan acceso los auditores sino también la ciudadanía en su conjunto.
5. Recomendamos que se realice un análisis de riesgos.
6. Recomendamos que se realice un estudio antes de encarar futuras elecciones que utilicen tecnología en la emisión del voto para evaluar alternativas tecnológicas de soportes documentales; en particular, la posibilidad de reemplazar la tecnología RFID, pasible de diversos tipos de ataques referidos en informes anteriores y su reemplazo por una alternativa que podría ser la impresión de códigos QR u otras.

   Para generar un identificador unívoco en cada soporte (que en el caso de la tecnología RFID viene "pregrabado" en el chip) será necesario un mecanismo aleatorio limpio y publicable. Dado que esto requiere del uso de algoritmos de generación de números pseudoaleatorios (PRNG) y las máquinas de voto, por su

naturaleza, carecen de la entropía necesaria para inicializar la generación en forma suficientemente aleatoria, podrá ser necesario agregar a la máquina hardware abierto de generación de números aleatorios (TRNG)[7].

En cualquier caso, deberán analizarse las alternativas respecto de procesadores, lectores y otros componentes necesarios, junto con el firmware, así como la longevidad y confiabilidad del soporte documental.

Sobre la funcionalidad

7. Recomendamos especificar que el sistema pueda operarse íntegramente sin la necesidad de la participación del proveedor del servicio; que el sistema incluya un componente que, debidamente configurado[8], permita generar o instalar los demás componentes, incluyendo la configuración e instalación de las máquinas de voto, de escrutinio y de transmisión de resultados[9]; esto debe poder hacerse aún cuando la configuración y operación se encomiende al proveedor, ya que es necesario para poder auditar y validar todos los pasos.
8. Recomendamos evaluar la inclusión de una instancia donde los partidos políticos puedan fiscalizar la información con la que se configuró el comicio (información sobre agrupaciones, partidos y alianzas, sus números, denominaciones, candidatos, fotos y logos, configuración de mesas, comunas y demás parámetros necesarios para la generación o instalación de las máquinas de votación, escrutinio y transmisión).
9. Recomendamos que el sistema provea un mecanismo para que el elector pueda verificar su voto de forma clara, sencilla y confiable.
10. Respecto del tratamiento del voto en blanco se recomienda que su posición sea aleatoria como sucede con el resto de los candidatos lo mismo que el espacio y forma que ocupa en la pantalla, teniendo en cuenta que, a diferencia del mecanismo de voto tradicional con boletas por partidos, el voto en blanco requiere de una acción y no de una omisión.
11. Recomendamos que el sistema le recuerde al elector que verifique su voto (independientemente de si la tecnología usada es un chip RFID, un código QR u otro mecanismo). Verificar el voto por parte de cada votante es uno de los factores *fundamentales* que ayudan a la confiabilidad y transparencia del proceso electoral.

7 Como ser el OneRNG (onerng.info)
8 para ello debe haber un manual suficientemente específico.
9 independientemente de si la máquina es la misma o no.

12. Recomendamos que el sistema genere el voto emitido en un soporte documental que permita realizar el escrutinio en forma manual, sin necesidad de utilizar ninguna tecnología para su lectura y conteo.
13. Recomendamos que, siempre que se siga utilizando este u otro sistema de emisión de voto o boleta electrónica, el control del padrón se continúe haciendo en forma manual (en papel) de modo de prevenir que exista la posibilidad de correlación masiva de votos con votantes, violando el secreto del sufragio.

Sobre las máquinas de votación

14. Recomendamos que todas las máquinas de votación y escrutinio dispongan de baterías que garanticen autonomía en su funcionamiento durante todo el proceso electoral; esto debe incluir la apertura de mesa, la votación de la totalidad del padrón de la mesa, el cierre de la mesa así como el escrutinio.
15. Recomendamos que si se necesita la custodia visual de elementos físicos (por ejemplo la tapa de acceso a los puertos y la lectora de DVD) se agreguen medios físicos para ayudar dicha custodia: tapa bien visible, de suficiente tamaño, colores, alarma visual y auditiva, etc.
16. Recomendamos que si hay tapas de acceso a dispositivos que deben ser utilizados durante el comicio (como ser la del lector de DVD), la misma no brinde acceso a dispositivos que no se utilicen (por ejemplo, puertos USB o Ethernet), que deberán tener tapas independientes y precintables (independientemente de que también tengan alarmas visuales o auditivas).
17. Recomendamos que los puertos que nunca deben usarse en la máquina en modo voto (USB, Ethernet, VGA) estén separados físicamente de los que sí se usan (lector de DVD, botón de encendido, auriculares). Los puertos que no se usan deberían estar físicamente bloqueados y ese bloqueo debería poder precintarse. Además, deberán estar completamente deshabilitados en el software.
18. Recomendamos que se haga un control de validez de la BIOS, el firmware de todos los componentes de la máquina de voto y el kernel del sistema operativo utilizando TPM o un mecanismo similar y verificable. El equipo debe verificar, cuando arranca, que todos esos componentes son confiables y no han sido alterados.
19. Recomendamos que durante el arranque de la máquina de voto se verifique la existencia y buen funcionamiento de todos los componentes críticos de hardware (e.g. TRNG).
20. Recomendamos que el mecanismo de validación del DVD desde la aplicación (en el menú del técnico o donde estuviere) utilice un hash seguro como ser

SHA512 y no uno inseguro como MD5 y que no se base en una lista de archivos previamente incluida en el DVD si no que arme la lista dinámicamente en base a la totalidad de los archivos efectivamente contenidos en el DVD.

21. Recomendamos que los paquetes del sistema operativo contenidos en el DVD de votación, una vez que se haya testeado y congelado el software, se mantengan en las versiones estables y no sean actualizados cuando aparezcan nuevas versiones a excepción de que estas nuevas versiones resuelvan vulnerabilidades que sean aplicables en el entorno del sistema de votación.

Sobre los servidores

22. Recomendamos que los servidores de recepción, consolidación y/o sumarización del escrutinio provisorio y de publicación de resultados, y su red estén diseñados de modo que puedan instalarse donde la autoridad lo considere, siempre que el lugar cumpla los requisitos necesarios para garantizar el funcionamiento y la seguridad de esas tareas; a su vez deben estar completamente definidos la arquitectura de red y de servicios de transmisión, deployment, seguridad, plan de contingencia, etc.

23. Recomendamos que todos los sistemas que requieran autenticación (carga, transmisión de datos, soporte de operaciones, monitoreo, etc) tengan mecanismos de autenticación confiables que vayan más allá de un usuario y una clave.

    Mínimamente, el esquema de autenticación con usuario y clave deberá tener un mecanismo complementario simple de claves de un solo uso (OTP) con tarjetas con las claves impresas, tarjetas de coordenadas o similares que deberán distribuirse a los usuarios.

Sobre los planes de contingencia

24. Recomendamos que los planes de contingencia se acuerden entre el proveedor del sistema y las autoridades correspondientes.

25. Los planes de contingencia deberán ser exhaustivos y contener instrucciones precisas acerca de cómo actuar ante cualquier eventualidad prevista.

26. Deberá haber planes específicos respecto del sistema de emisión de voto, del sistema de transmisión de datos, del sistema de totalización y del sistema de publicación de resultados.

Sobre los procedimientos y la capacitación

27. El sistema deberá contar con manuales de procedimientos para cada uno de los usuarios del mismo (autoridades de mesa, delegados judiciales, técnicos, operadores, etc).
28. La definición de los procedimientos deberá ser exhaustiva y contener recomendaciones y precisiones que aseguren la confiabilidad del sistema (ver el Anexo II del Informe N° 2 de auditoría).
29. Se deberán preparar planes de capacitación para cada uno de los usuarios del sistema, incluidos los ciudadanos en su calidad de electores. En el caso de los planes masivos (para delegados judiciales, autoridades de mesa y especialmente para electores) se deberá prever un mecanismo de capacitación de capacitadores.

Conclusiones

El proceso electoral de la Ciudad de Buenos Aires, con las PASO utilizando el sistema tradicional de boletas por partidos y las elecciones generales y la segunda vuelta utilizando la Boleta Única Electrónica fue exitoso y sin mayores sobresaltos.

El sistema en general, tanto el hardware, como el software y, particularmente, los procedimientos funcionaron en forma correcta y esta auditoría entiende que la ciudadanía pudo expresar su voluntad.

Una de las características del sistema de votación y escrutinio auditado es que es comprobable físicamente y la voluntad de los electores se puede verificar en forma completamente manual, sin la intervención del sistema. Esto es importante en casos extremos (aunque improbables) de fallas generalizadas o ante un requerimiento de fiscalización por parte de las agrupaciones políticas que se considere admisible.

Esta auditoría considera que el mayor resguardo del sistema consiste en los mecanismos de control existentes que son externos a la solución tecnológica, lo cual es facilitado por los procedimientos definidos en el sistema auditado.

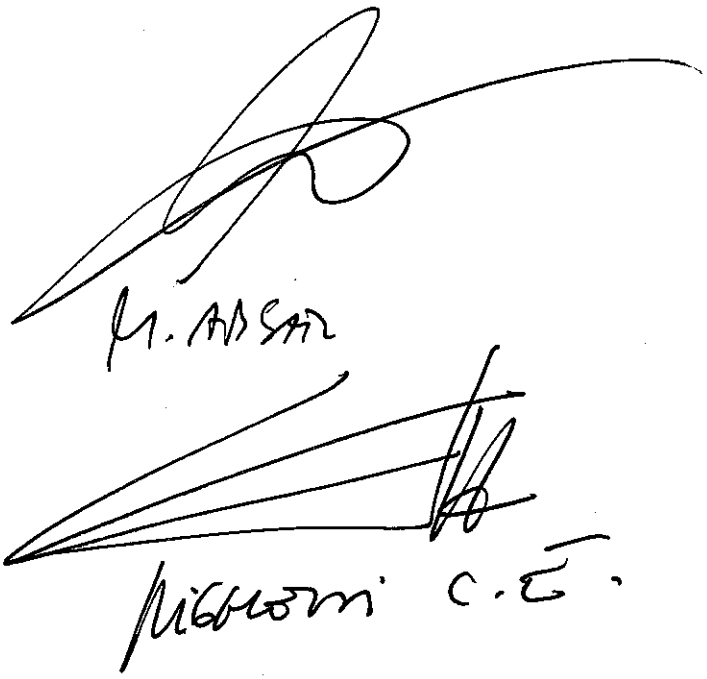
Los principales custodios de los comicios siguen siendo tanto las autoridades de mesa y los delegados del Tribunal, como los fiscales de las agrupaciones políticas y los mismos electores.

Más allá de que las jornadas electorales transcurrieron sin inconvenientes y el sistema funcionó correctamente, todo el proceso de adopción, adquisición y auditoría fue hecho con demasiado apuro y con algún grado de improvisación, impidiendo que los procesos en general y la auditoría y la capacitación en particular, sean más efectivos.

Un proceso de adquisición de servicios para llevar adelante comicios en un distrito completo utilizando tecnología desde la emisión del voto requiere de tiempos mucho más largos y mecanismos más aceitados de comunicaciones entre todos los involucrados (auditores, organismo responsable de llevar a cabo los comicios, organismo responsable de auditar los sistemas -especialmente cuando *no* es el mismo organismo que lleva adelante los comicios y las contrataciones, empresa contratista, etc).

En la sección Recomendaciones se aconsejan tiempos mínimos para llevar adelante estas tareas que están muy por encima de los tiempos con los que contó esta auditoría que comenzó a fines del mes de febrero para auditar sistemas para actos eleccionarios a llevarse a cabo a fines de abril y mediados de julio del mismo año.

Si se fuera a utilizar este sistema u otro que incluya tecnología desde la emisión y el escrutinio en la mesa (es decir, tecnología cuyos usuarios son el conjunto de los electores y las autoridades de mesa), se recomienda fuertemente seguir las pautas y recomendaciones contenidas en este informe para poder contar con una auditoría mucho más profunda y ayudar a garantizar la confiabilidad del acto eleccionario.



Anexos

Anexo I – Auditoría del código fuente

Objetivos

La tarea de auditoría sobre código fuente tiene como objetivo detectar:

1. *fallas funcionales*, cuando el programa no hace lo que debería hacer;
2. *vulnerabilidades*, cuando existe alguna manera de forzar a alguna parte del programa a hacer cosas distintas a las se espera según sus especificaciones;
3. *debilidades*, cuando alguno de los módulos está programado de tal manera que si se hacen cambios en otros módulos o reestructuración de código o cambios en la configuración del programa los módulos débiles deben modificarse también; cuando la cantidad de módulos débiles es grande cada modificación implica una cadena de actualizaciones que deben realizarse a mano y que cuando no se hacen pueden dejar al programa con fallas o vulnerabilidades directas. Algunas debilidades son:
   a) vulnerabilidad indirecta (que no pueden ser explotadas salvo que se encuentren otras que combinen con ella);
   b) alto nivel de acoplamiento con otros módulos;
   c) tiene datos de configuración incluidos dentro del código.

Auditar el código fuente de un sistema no puede garantizar o verificar que el mismo cumpla con los requerimientos solicitados, aún cuando la especificación de los objetivos y requerimientos del software sea formal y perfecta.

Requerimientos y especificaciones

Es importante a la hora de analizar el código fuente (y los demás aspectos funcionales y de seguridad del sistema) conocer los requerimientos y que estén definidas las especificaciones de todos los módulos. Cuando los requerimientos no están escritos, o se basan en distintas fuentes (ejemplo: artículos de una ley, resoluciones, acordadas, etc) o las especificaciones no están o no son exhaustivas, el análisis del código fuente se hace sobre supuestos, esto puede hacer que se sobrestimen o subestimen ciertas observaciones, o que pueda haber error en el análisis del mismo.

Metodología de desarrollo y convenciones de codificación

La práctica profesional desarrolló un sinnúmero de metodologías de desarrollo, establecimiento de buenas prácticas, convenciones, etc. Es necesario que los equipos de trabajo adopten una metodología unificada, que esa metodología esté escrita (ya sea porque es una metodología propia o porque se ha adoptado una existente, en cuyo caso debe estar claro el link a la versión adoptada) y sea explícita en la documentación presentada a la auditoría. Cuando por alguna razón, para ciertos módulos se hagan variaciones o se cambie la metodología adoptada, también debe explicitarse en la documentación.

Pruebas del sistema

Las metodologías modernas de desarrollo de sistemas incluyen las pruebas automatizadas. Las pruebas automáticas tienen la ventaja, con respecto a las pruebas manuales, de que pueden ser corridas innumerables veces en un lapso corto de tiempo; esto permite probar todo el sistema cada vez que se introducen modificaciones al mismo. Las pruebas automáticas no son otra cosa que un programa que prueba otro programa y por lo tanto necesita ser escrito por un programador[10] trabajando en equipo con los profesionales que diseñaron el sistema y/o escribieron los requerimientos y especificaciones.

La calidad de las pruebas se mide utilizando varios aspectos, uno de ellos es la cobertura de código, y para que un sistema sea auditable, es fundamental que tenga una amplia cobertura. La cobertura de código mide qué porcentaje de las líneas de código son utilizadas durante el conjunto total de pruebas (por ejemplo si la cobertura es el 90% significa que hay un 10% de las líneas de código que nunca son usadas durante las pruebas). Un sistema puede tener una alta calidad de pruebas aún cuando la cobertura no sea del 100%, si las partes del código que no cubren las pruebas están debidamente identificadas y debidamente justificada su exclusión de las pruebas.

Tener pruebas automáticas de código no suplen las pruebas manuales con distintos tipos de usuarios reales. Estas pruebas manuales deben también documentarse en un plan de pruebas que tiene que estar disponible para la auditoría. Los auditores tienen que estar invitados a realizar o presenciar cualquiera de las pruebas del plan.

10 Cuando no se cuenta con programadores especializados, el costo de las pruebas puede parecer muy alto.

Separación de la configuración y el código

Atributos de la estructura de los metadatos

Muchas veces se encuentran dentro del código fuente, ya se explícita o implícitamente, valores que deberían estar fuera del código fuente y dentro de las configuraciones. Por ejemplo no debería haber dentro del programa algo como:

```
where nlevel(id_ubicacion) = 3
```

para identificar un local o escuela (suponiendo que las ubicaciones son una jerarquía), en vez de ese número 3 debería haber una constante `MAX_NLEVEL_UBICACION` (si el programa no permite ni permitirá más niveles de ubicación que 3) o una función `max_nivel_ubicacion()` si se le permite al usuario del sistema definir en cuántos niveles organizará la logística en cada elección. Usar una función (o una variable) es más flexible e imprescindible cuando el programa permite elegir la cantidad de niveles. Cuando el sistema tiene una cantidad fija de niveles es preferible una constante con nombre antes que un número suelto. Así, si más adelante se decide generalizar el programa para permitirle al usuario elegir otra cantidad de niveles, quedan mejor señalados los puntos del programa que deben modificarse. Si por el contrario se dejó una constante (el número 3 en este caso), cualquier 3 es susceptible de cambios. Peor aún es la situación cuando el número no es directo:

```
where nlevel(id_ubicacion) <= 2
```

donde ese 2 representa los niveles de localización o jerarquías que incluyen otras, en ese caso sería mejor:

```
where nlevel(id_ubicacion) <= MAX_NLEVEL_UBICACION - 1
```

Todavía esto puede no ser suficientemente genérico porque, por ejemplo, en un operativo nacional la cantidad de niveles en que se divide la logística podría depender de la provincia donde se esté (o del departamento dentro de la provincia) en ese caso es mejor indicar a cada ubicación si es una hoja del árbol de categorías:

```
where es_hoja(id_ubicacion) = true
```

Casos especiales

A veces ciertos elementos de una clase de objetos se comportan distinto que otros elementos de la misma clase. Por ejemplo si la forma de desplegar la opción de las distintas categorías de candidatos a elegir se diferencian de modo que para la categoría jefe de gobierno es un modo y el resto otro, en vez de programar algo como:

```
if candidato.cod_categoria == "JEF":
    secundarios = candidato.secundarios
```

es mejor independizar la diferencia de comportamiento del código que se le asignó a la categoría. Una buena regla para saber si la constante que se está agregando es una configuración de caso especial (y por lo tanto debe programarse de otra manera que no incluya el código de la categoría dentro del programa) es preguntarse ¿qué debe decir el manual? "*Si desea que una categoría tenga candidatos secundarios póngale de código «JEF»*" parece demasiado específico (en especial porque la categoría podría llamarse de otro modo o porque se podrían llegar a querer dos categorías que tengan candidatos secundarios; aún cuando parezca que eso es imposible, con el mismo criterio que un Jefe puede tener un Vicejefe y no considerárselo como una lista de candidatos del tipo de lista de legisladores, podría ocurrir que en una comuna se elija el defensor de la comuna que venga con su vicedefensor).

Orden en que se despliega la información

Parte de la configuración que debería permitirse definir al usuario administrador tendría que ser el orden en que se despliega la información en las distintas pantallas y/o planillas; el sistema debe proporcionar esta funcionalidad de manera flexible y cómoda.

No alcanza con ordenar alfabéticamente[11] por el campo "nombre", (porque en el caso de las PASO, por ejemplo el orden en el acta de cierre las agrupaciones hay que ordenarlas por partido/alianza y luego por candidato). Puede parecer una solución flexible agregar un campo "orden" en cada entidad configurable, de este modo el usuario administrador tendría que numerar todo lo que deba ordenarse (locales de votación, categorías, candidatos, etc), esto es útil si el orden que se le quiere dar a los elementos es un orden distinto al que se podría lograr especificando criterios de ordenamiento basando en campos ya definidos (ejemplo: lema, sublema) y agrega una incomodidad en esos casos. Una posible alternativa a evaluar podría ser:

Para definir una forma de ordenar una entidad agregar un campo "orden" a los elementos de la entidad y tener una "lista de campos que definen la ordenación". Si sólo se desea usar el campo orden, en la lista se pone "orden". Si se desea ordenar por lema y dentro de cada lema ordenar por fecha de inscripción, y ese dato no se carga en la configuración del sistema, se puede poner en el campo "orden" el orden dentro del lema y en la lista de campos que definen la ordenación se indica primero lema y luego orden.

11 considerando el orden alfabético que ordena los números en forma natural considerando todas sus cifras y no dígito a dígito.

Hay que prestar atención porque hay ciertas circunstancias donde el orden en que se despliega la información depende del lugar del programa donde eso se está produciendo, por ejemplo en la pantalla para elegir el orden debe ser aleatorio y sortearse y en las planillas debe estar fijo.

Caso general

Cada vez que se introduzca un dato o una constante debe pensarse si eso no reduce o condiciona innecesariamente el alcance del sistema o la configuración del mismo.

Aplicaciones basadas en un *viewport* HTML

A veces las aplicaciones diseñadas para correr en una sola máquina están estructuradas en un esquema similar al desarrollo web, con las siguientes diferencias:

1. el backend y el frontend corren dentro de la misma máquina;
2. no hay navegador (no se ve el cuadro de la URL, ni los botones para retroceder, cancelar la navegación, etc);
3. la pantalla inicial (correspondiente a la primer página desplegada) es decidida por el *backend* y para ir a otras pantallas deberá interactuarse con los elementos desplegados en pantalla (botones, vínculos, *scripts*, etc);
4. a veces las validaciones se hacen solamente en el *frontend* y no en el *backend*, bajo el supuesto de un ambiente controlado; lo que hace que aumenten las debilidades del código que se corre en el *frontend*.

Inyección de código

Se llama inyección de código a un tipo de vulnerabilidad del software que consiste en introducir (a través de un parámetro o canal de comunicación entre el usuario y el software, o entre distintas partes del software cuyo canal de comunicación se pueda interceptar) instrucciones que permitan ejecutar código fuente que no es original del programa. Según el tipo y estructura del programa, el código inyectado podría lograr borrar, alterar u obtener información de la base de datos, del sistema de archivos o de variables internas del programa; o cambiar el comportamiento futuro del programa.

El código vulnerable puede estar tanto en el *frontend* (en especial en aplicaciones basadas en HTML) como en el *backend*.

Ejemplo

```
$("#datos").html("<h3>Obs: " + input1.value + "</h3>");
```

Esa sección de código JavaScript coloca una fracción de código HTML dentro del elemento "datos" de la pantalla pudiendo introducir código inyectado. Por ejemplo si la variable `input1` refiere a un elemento *input* en pantalla, el usuario pudiera escribir directamente código.

Otro ejemplo: sabiendo que hay una función JavaScript llamada `tienePermiso` que se usa para darle acceso al usuario a las distintas partes del programa, el usuario podría escribir en el *input*:

```
<script>tienePermiso=function(){ return true; }</script>
```

para reemplazar la función `tienePermiso` por otra que siempre devuelve que sí lo tiene. El código que se coloca entre `<script>` y `</script>` puede ser cualquiera y si el *backend* confía en el *frontend* se pueden invocar las funciones del JavaScript destinadas a comunicarse con el *backend* explotando dicha confianza.

Esto se podría evitar cambiando la función `html` (que introduce código) por la función `text` (que introduce texto), así:

```
$('#datos').append($('<h1>').text('Obs: '+input1.value));
```

O, en el caso de que se desee permitir que el usuario ingrese algunos comandos de formato (por ejemplo <b> para poner negritas), se invoque antes a una función validadora que verifique que se ingresen solo los códigos permitidos[12].

Acciones correctivas del caso general

Para evitar este tipo de vulnerabilidad hay que prestar especial atención a las funciones que reciben código (HTML, SQL, JavaScript, shell del sistema operativo, Python, etc), el código que se envía no debe incluir los datos dentro del código (salvo que se validen en forma completa y perfecta). La mayoría de las veces es preferible utilizar las alternativas que permiten separar el código de los datos.

12 tener en cuenta que `<script>` no es la única manera de introducir código JavasSript, también se puede incluir en los atributos de eventos para casi cualquier *tag* (e.g. `<div onclick=...>`) incluso en lugares donde se pueden especificar URLS (porque se podría escribir `javascript:fun...`). Por eso es preferible, cuando se va a usar la función `html` o `innerHTML` prohibir en general cualquier texto que contenga "<" y tener una lista de *tags* permitidos (por ejemplo <b>, <i>, etc).

Ejemplos:

	en vez de	*es preferible*
jQuery	`$(id).html('<b>'+valor+'</b>');`	`$(id).append($('<b>').text(valor));`
SQL	`query("delete … where x="+valor)`	`query("delete … where x=$1", [valor])`
S.O.	`spawn("proc "+valor)`	`spawn("proc", [valor])`

Cuando no se disponga de una función que permita separar el código de los datos debe utilizarse siempre una función que quite o "escape" los caracteres que permiten la inyección de código (cosa a veces compleja; siempre es preferible utilizar las librerías estándar para hacerlo).

La función eval

La inyección de código no ocurre solamente cuando se pasan programas o instrucciones en cadenas de texto a módulo programados en otro lenguaje, también ocurre dentro del mismo lenguaje, por ejemplo usando la función `eval`[13]. Su uso tiene que evitarse en la medida de lo posible. Por ejemplo si desde el *backend* se quiere enviar al *frontend* la indicación de la próxima acción que debe realizar el *frontend*, una posibilidad es enviar una cadena de texto con la llamada a la función:

```
function run_op(proxima_accion){
    eval(proxima_accion); // desaconsejado
}
```

Es preferible pasar por separado los datos del nombre de la función que se desea ejecutar. De esta manera ya no es necesario usar `eval`:

```
function run_op(operacion, data){
    func = window[operacion] // con cuidado[14]
    data = JSON.parse(data);
    func(data);
}
```

13 En JavaScript, además de `eval`, hay varias funciones que aceptan cadenas de texto con código JavaScript (por ejemplo `setTimeout`). Otros lenguajes tienen otras funciones o estructuras, por ejemplo EXECUTE en PL/pgSQL.

14 esto es mejor que `eval` pero también tiene el problema de que podría ejecutarse cualquier función global.

El nombre de la función debe usarse para indexar un mapa de funciones permitidas porque, al estar el *frontend* y *backend* se sabe de antemano cuáles son las funciones que se podrían necesitar (en especial cuáles son seguras):

```
function run_op(operacion, data){
    func = funciones_permitidas[operacion];
    data = JSON.parse(data);
    func(data);
}
```

Otro ejemplo de posible uso de funciones tipo `eval` ocurre cuando se desarrollan sistemas configurables que le permiten al usuario administrador definir criterios de validación o de avisos. Por ejemplo el sistema podría tener un lugar donde configurar cuándo debe advertir que se está por cerrar una mesa con pocos votantes. Se le podría permitir al usuario escribir expresiones del tipo:

```
votos_contados / total_padron < 0.1  &&  total_padron > 20
```

Para evaluar esa expresión dentro del programa se debe usar una función que evalúe expresiones. Es razonable usar `eval` (especialmente si la alternativa es programar una función evaluadora) pero tomando la precaución de validar la expresión para que sólo contenga funciones y variables permitidas. Hay que tener una lista de elementos permitidos (funciones y variables) y controlar que cada palabra de la expresión esté dentro de la lista de elementos permitidos.

```
function evaluar_expresion(expresion, funciones_ok, variables_ok){
    var regExpVarFun=/\b([a-zA-Z_]+)(\s*\()?/;
    expresion.replace(regExpVarFun,function(_, name, isFun){
        if(!(isFun?funciones_ok:funciones_ok)[name]){
            throw new Error(" inválido en expresión: "+name)
        }
    });
    return eval(expresion); // la expresión se verificó arriba
}
```

Manejo de excepciones y recuperaciones ante errores

En casi todos los programas pueden ocurrir situaciones excepcionales. Por ejemplo:

1. el disco donde se debe escribir un archivo temporal está lleno (no tiene espacio libre);
2. la conexión entre el *frontend* y el *backend* tuvo un corte;
3. se llenó temporariamente un *buffer* intermedio de conexión;
4. un campo de texto tiene algún carácter inválido o no está codificado como se espera (utf8 en vez de ANSI, etc).

Algunas de estas situaciones pueden evitarse (por ejemplo los caracteres inválidos) no permitiendo el ingreso desde el teclado y controlando la información que se carga masivamente cuando se inicia la aplicación o se instala la base de datos. Otras veces es imposible evitarlas o preverlas (por ejemplo que se llene el recurso sobre el cuál se está escribiendo, o que ocurra un error físico).

Los lenguajes modernos utilizan el mecanismo de manejo de excepciones para facilitar la tarea del control de errores y situaciones excepcionales. Aún usando esos lenguajes no todas las funciones de librería utilizan esos mecanismos, algunas informan los errores en forma directa[15].

Hacer caso omiso de los errores (o situaciones excepcionales) que pueden ocurrir al ejecutar una función es una tentación para algunos programadores. Pensar en "esto no puede pasar" o "es tan raro que no vale la pena considerarlo" pueden llevar a que los programas, en esos casos, tengan comportamientos inapropiados.

Regla de Oro: **No permitir que el programa continúe si pudo haber ocurrido un error o una situación excepcional.**

Los puntos del programa a los que hay que prestar atención son:

1. las funciones de librería que según el manual puedan devolver error o lanzar excepciones;
2. las funciones de librería que no tienen documentación exhaustiva;
3. las expresiones numéricas que tengan divisiones cuyo denominador sea variable (porque puede ser cero) o que puedan excederse de la precisión aceptada;
4. las expresiones que incluyen conversiones de tipos;
5. la adquisición de recursos;
6. las llamadas a módulos externos (sistema operativo, base de datos, *backend*, *frontend*) o cualquier llamada que utilice explícita o implícitamente un canal de comunicación;
7. las funciones programadas que usen en cualquier función a la que haya que prestarle atención.

En cada punto a prestar atención deben considerarse las siguientes acciones:

1. Cada vez que se llame a una función que devuelva una condición de error en forma directa (o a través de un *callback*) hay que controlar explícitamente que no se haya producido un error y actuar en consecuencia. **Ignorar los errores puede conducir a un problema grave**;

15 la forma directa, cuando la función es asincrónica, consiste en llamar a una función *callback* a la que se la pasa como parámetro la condición de error.

2. Cuando se conozca alguna circunstancia en la que un error o una situación excepcional puedan ser normales (o toleradas), debe asegurarse de admitir *solamente* ese tipo de excepciones y no estar ignorando errores o situaciones excepcionales de otros tipos;
3. Cuando se manejen excepciones debe cuidarse de admitir o actuar siendo lo más preciso posible respecto del tipo de excepción.

Ejemplos de errores comunes

1. Sin control de errores:

```
$.ajax(url, {async: false,
    success: function(data, textStatus, jqXHR){
        template = jqXHR.responseText;
     }
});
```

Si se produce un error en la llamada ajax, o hay un problema durante la comunicación con el *backend* o con la recepción del mensaje, el programa ignora el problema sin registrar o avisar. Falta una función útil para el evento error.

2. Captura amplia de la excepción:

```
v_porc := round(v_parte::numeric / v_total::numeric * 100 , 1)
:: text;
exception
    when others then
        v_porc := '-';
```

En este ejemplo (PL/SQL) el programador sabe que al principio no se conoce `v_total` y viene en cero, por lo tanto esa división podría lanzar una excepción, en ese caso quiere devolver un guión. El problema es que se están capturando *todas* las excepciones posibles, por ejemplo si el texto que viene en `v_parte` o `v_total` no se puede convertir a un número. En este caso se podría haber puesto un "`if`" explícito para detectar cero en el denominador o utilizar el código de la excepción en la cláusula "when" (`when division_by_zero then`), de ese modo el comportamiento es el deseado en la condición prevista. En los lenguajes (como JavaScript) donde no se puede especificar el tipo de la excepción capturada lo que debe hacerse es relanzar la excepción cuando no sea el caso esperado (ya sea comparando con el tipo de la variable `err` o con el código guardado dentro de la variable `err`):

```
catch(err){
     if(err.code != "22012") throw err;
     if(typeof err !== TypeError) throw err;
}
```

3. Descarte explícito del error:

Algunas construcciones o formas de invocación de funciones o instrucciones pueden descartar los errores. Es el caso de la instrucción `PERFORM` de PL/pgSQL (que explícitamente descarta los resultados devueltos por la función llamada, sean errores o valores válidos). Algunos programadores utilizan `PERFORM` como manera de invocar funciones que normalmente no devuelven valores sin analizar el problema del enmascaramiento de eventuales excepciones.

```
PERFORM crear_usuario(p_username, …
```

¿Qué debe hacer el programa en caso de errores o situaciones excepcionales?

1. Lo peor que se puede hacer es caso omiso (o ignorar el error). Prever todas las situaciones excepcionales que podrían ocurrir y evaluar si el programa no va a hacer cosas no esperadas en todos esos casos es muy difícil.
2. Una alternativa es interrumpir (abortar) el programa (o el módulo) mostrando un cartel indicando que ocurrió un error. Esto podría parecer muy drástico, pero dejar que el programa haga algo imprevisto puede ser un problema más grave. En general esta es la mejor situación, salvo que haya una buena justificación para permitir que el programa esté en una situación no prevista donde no se sepa cómo puede reaccionar.
3. En caso de interrumpir hay que decidir cuánta información se muestra. Por un lado cuanto mejor se sepa en qué punto del programa se originó el problema es

TSJ
315

más fácil corregir el problema, por otro lado mostrar datos internos al usuario no puede ser una regla general porque la información interna podría contener información sensible o no permitida para el usuario.

4. Otra alternativa a veces usada es permitir que el programa continúe y guardar un registro de las operaciones. Eso no es una solución general porque en algunos sistemas (ejemplo: la máquina de votar) no está permitido guardar registros de ningún tipo. Aún en los sistemas donde sí se puede tiene que haber un control serio sobre esos registros.
5. Hacer algo distinto en la etapa de testing, por ejemplo mostrar más datos (incluyendo las variables internas) o interrumpir (en vez de dejar seguir) puede ser una opción útil pero hay que asegurarse de que en la etapa de uso en producción los cambios no son tales que el testing no sirva (en particular hay que testear también "en modo producción").

Como el tema de errores y situaciones excepcionales es un tema delicado, para que el sistema sea auditable, las decisiones que se hayan tomado al respecto tienen que ser explícitas y estar incluidas dentro de la documentación. En especial deben justificarse los usos no convencionales, las situaciones que se analizó que se está seguro que no ocurrirán, los riesgos que se aceptar tomar y las acciones planificadas. O sea cada vez que se encuentre en el código alguno de estos casos debe estar justificado el análisis que determina que no es una debilidad del código.

Recibido el 14/8/15 a las 13.0 horas.

Dr. Roberto Asorey
Prosecretario Mayor
Secretaría Judicial en Asuntos Originarios
Tribunal Superior de Justicia
Ciudad Autónoma de Buenos Aires