Capitalistic Console Cracking

RMS – CAN CRAFTILY CIRCUMVENT CERTAIN CRAPPY CERTIFICATE CHECKSUMS

The eighth generation of video game consoles (both stationary and portable) is fast approaching, and as control schemes diverge, and franchises swap bases (really, FF XIII?), let's take a look at the past machines and the machinations of one of their manufacturers.

Twilight Bug

In a classical buffer overflow and stack smashing exploit, early versions of the Wii firmware were viable to a game implementation dependent memory attack: Manipulating a saved game of The Legend of Zelda: Twilight Princess to include a custom name for the protagonist's horse (canonically named Epona) allowed the execution of arbitrary code. The originally included snippet would automatically attempt to run "boot.dol" or "boot.elf" from the root of the SD card slot. This made it possible to install custom channels and bootloaders, opening the doors for Homebrew fans everywhere.

In truly good news for programming methodology, the original implementation of the exploit was written in such a portable and reusable manner, that its adaptations and forks continue to thwart Nintendo's counter-hacking efforts.

Be careful when googling for further information: The "Twilight Bug" is apparently also a vicious creature whose bite may turn you into an avid fan of Stephanie Meyer's "literary works" [sic].

Trucha bug

In another exploit that might delight the avid C programmer, the so-called "Trucha bug" happened during the authentication stage of executables. As for many console generations before, a system would only execute programs that were correctly signed by the manufacturer – in SNES days, this meant a physical chip on the cartridge; on the Wii, any game disk and digital download has been signed using the SHA-1 algorithm and an official Nintendo certificate. Unfortunately, the implementation of the SHA hash comparison has a serious flaw. See if you catch it in the pseudo C code below (also available at [1]):

```
int verify_cert (struct rsa_cert cert)
{
  char *cert_hash=SHA1(cert.metadata +
                       cert.content_hash);
  char *sig_hash=rsa_decrypt(
                 cert.rsa_signature,
                 cert.key_id);

  if (strncmp(cert_hash, sig_hash,
        SHA1_LENGTH) == 0)
  {
     return CERT_OK;
  } else {
     return CERT_BAD;
  }
}
```

The function used here to compare the two strings cert_hash and sig_hash is strncmp, "String-N-Compare". This function returns 0 if the two strings are either equal or if their first N characters are equal (here N is SHA1_LENGTH). The problem here is the internal representation of SHA hash strings – which in fact may contain the NULL byte '\0', signifying the end of C strings. Strncmp interprets this as the end of a string and thus will immediately return with the result so far.

This means that instead of having to brute-force a hash of SHA1_LENGTH, one can just randomly generate SHA hashes that have a NULL byte fairly early, and then brute-force for hash collisions, which immensely reduces the computational complexity involved to mere seconds.

Theories exist that this was just a plain programming mistake, as the function "memcmp" has nearly the same interface, and will in fact not abort on NULL bytes. However, from a strategic point of view, these errors might have been deliberate on the manufacturer's part to make their console easy to crack.

The Theory

But why should Nintendo purposely forego profit? The crux lies within the duality of income in the console industry: Firstly, the manufacturers directly profit from each sold hardware unit, controller, and physical extension. Secondly, they sell their SDKs and official certification to video game developers, who are thus indirectly responsible for part of the income. Out of the three big competitors Microsoft, Sony, and Nintendo, the latter is the only one who has – starting on release day – actually made money from each sold console unit; the first two instead banking on SDK and license sales.

This makes the supposed strategy much more reasonable: No matter if your operating system has been exploited, you will still sell hardware even to pirates who will not buy the licensed games. The only downside lies within the alienation of the video game developers, but Nintendo took that risk, successfully betting on the success of their innovative control method, which would motivate developers to continue to produce Wii games.

Conclusion

This is certainly only a theory; but keep this history in mind for the upcoming console generation. At this point it should also be said that both bugs mentioned above have been fixed in firmware upgrades; derivatives do, however, still exist, and the cracking community is always one step behind Nintendo, finding new exploits wherever they look. ⌥

Links

[1] http://wiibrew.org/wiki/Signing_bug

So selbstverständlich wie unsere Software.

Softwarelösungen sind dann gut, wenn man sie gar nicht wahrnimmt. Wenn im E-Banking das Login ganz sicher und reibungslos funktioniert. Wenn Sie Ihre Prozesse via Mausklick steuern können. Wenn Ihr Enterprise-Portal jedem Kunden das passende anbietet. Unspektakulär und ohne Probleme, wie Lichteinschalten. Interessiert? Drücken Sie die Telefontasten und reden Sie – selbstverständlich mit uns. AdNovum Informatik AG, Röntgenstrasse 22, 8005 Zürich, Telefon 044 272 6111. www.adnovum.ch

High-End Software & Security Engineering

AdNovum